IN THE CLAIMS:

1 1. (Currently Amended) A method for converting a file access data structure from a first

2 endianness to a second endianness by a processor, the method comprising the steps of:

3 identifying, from a descriptor look up table, a series of actions to perform on ele-

4 ments of the file access data structure, where the series of actions include at least one of

5 converting, copying, or linking; and

6 performing the identified series of actions on the elements of the file access data

7 structure to convert the file data structure from the first endianness to the second endian-

8 ness.

1 2. (Previously Presented) A method of converting elements of a file access data structure

2 from a first endianness to a second endianness by a processor, the method comprising the

3 steps of:

4 determining if the file access data structure is a critical path data structure;

5 converting, in response to the file access data structure being a critical path data

6 structure, the elements from the first endianness to the second endianness using a set of

7 specific code functions;

8 converting, in response to the file access data structure not being a critical path

9 data structure, a header of the file access data structure from the first endianness to the

10 second endianness using a second set of specific code functions; and

11   calling a byte swapping engine to convert selected elements of the file access data

12   structure from the first byte order to the second byte order.

1   3. (Original) The method of claim 2 wherein the file access data structure further com-

2   prises a direct access file access data structure.

1   4. (Currently Amended) A file system for converting elements of a file access data struc-

2   ture from a first endianness to a second endianness, the system comprising:

3   an input buffer, the input buffer storing the file access data structure with the first

4   endianness to be converted;

5   a byte swapping engine, the byte swapping engine operative interconnected with a

6   descriptor table, with the descriptor table listing a series of actions to perform when con-

7   verting the file data structure from the first endianness to the second endianness, where

8   the series of actions include at least one of converting, copying, or linking; and

9   an output buffer, the byte swapping engine placing the file access data structure

10   with the second endianness in the output buffer after conversion.

1   5. (Original) The system of claim 4 wherein the descriptor table further comprises a set

2   of entries describing various file access data structures, each entry further comprising a

3   size field and an operation field.

1   6. (Original) The system of claim 4 wherein the file access data structure further com-

2   prises a direct access file access data structure.

1   7. (Previously Presented) A method for converting a data structure from a first byte order

2   to a second byte order by a processor, the method comprising the steps of:

3   reading an element entry from a descriptor table;

4   performing an action on an element of the data structure, the action being defined

5   in the element entry read from the descriptor table to convert the data structure from the

6   first byte order to the second byte order; and

7   placing the element in an output buffer.

1   8. (Original) The method of claim 7 wherein the step of performing an action on an ele-

2   ment further comprises the step of copying the element from an input buffer to the output

3   buffer.

1   9. (Original) The method of claim 7 wherein the step of performing an action on an ele-

2   ment further comprises the step of byte swapping the element.

1   10. (Original) The method of claim 7 wherein the element entry of the descriptor table

2   further comprises a field describing a size of the element and a field describing an action

3   to be performed.

1   11. (Original) A file server for use in a network storage environment, the file server

2   comprising:

3   a byte swapping engine, the byte swapping engine performing a defined operation

4   on each of a plurality of elements of a file access data structure.

1   12. (Original) The file server of claim 11 wherein the file server further comprises a de-

2   scriptor look up table, the descriptor look up table having a plurality of entries, each of

3   the plurality of entries associated with a specific file access data structure.

1   13. (Original) The file server of claim 12 wherein each of the plurality of entries further

2   comprises a plurality of elements, each of the elements having a size field and an opera-

3   tion field.

1   14. (Original) The file server of claim 13 wherein the defined operation is defined by the

2   operation field of the entry associated with the file access data structure.

1   15. (Previously Presented) A computer-readable medium, including program instructions

2   executing on a computer, for converting elements of a file access data structure from a

3   first endianness to a second endianness, the method comprising the steps of:

4   determining if the file access data structure is a critical path data structure;

5

5   converting, in response to the file access data structure being a critical path data

6   structure, the elements from the first endianness to the second endianness using a set of

7   specific code functions;

8   converting, in response to the file access data structure not being a critical path

9   data structure, a header of the file access data structure from the first endianness to the

10   second endianness using a second set of specific code functions; and

11   calling a byte swapping engine to convert selected elements of the file access data

12   structure from the first byte order to the second byte order.

1   16. (Currently Amended) A method for ~~converting~~ processing elements of a file access

2   data structure from a first endianness to a second endianness by a processor, the method

3   comprising the steps of:

4   determining a type of the file access data structure, where the type of the file ac-

5   cess structure is the first endianness;

6   processing, in response to the file access data structure of being of a first type, the

7   file access data structure along a first processing path; and

8   processing, in response to the file access data structure being of a second type, the

9   file access data structure along a second processing path, where the data structure of the

10   second type is the second endianness.

1   17. (Currently Amended) The method of claim 16 wherein the first type further com-

2   prises a critical path data structure, where the critical path data structure includes com-

3   monly utilized data structures.

1   18. (Original) The method of claim 16 wherein the first processing path further com-

2   prises a set of specifically coded functions.

1   19. (Original) The method of claim 16 wherein the second processing path further com-

2   prises a byte swapping engine.

1   20. (Currently Amended) A method for converting a data structure by a processor, com-

2   prising:

3   calling a byte-swapping engine;

4   providing a file access data structure as input to the byte-swapping engine;

5   providing a descriptor look up table to the byte-swapping engine;

6   identifying, from the descriptor look up table, a series of actions to perform on

7   elements of the file access data structure in order to swap bytes of the file access data

8   structure from a first endianness to a second endianness, where the series of actions in-

9   clude at least one of converting, copying, or linking; and

10   performing the identified series of actions on the elements of the file access data

11   structure to convert the file access data structure.

7

1 21. (Previously Presented) The method as in claim 20, further comprising:

2   using as the file access data structure a file having Direct Access File System

3 (DAFS) protocol.

1 22. (Currently Amended) The method as in claim 20, further comprising:

2   determining if the file access data structure is a critical path data structure, where

3 the critical path data structure includes commonly utilized data structures, and if ~~it~~ the file

4 access data structure is a critical path data structure ~~is~~, perform byte swap operations us-

5 ing specific code functions.

1 23. (Currently Amended) The method as in claim 20, further comprising:

2   determining if the file access data structure is a critical path data structure, where

3 the critical path data structure includes commonly utilized data structures, and if ~~it is not~~

4 the file access data structure is not a critical path data structure, perform byte swap opera-

5 tions on a data structure header.

1 24. (Previously Presented) The method as in claim 20, further comprising:

2   swapping bytes of the data structure as needed, in response to swapping bytes of

3 the file access data structure.

1 25. (Currently Amended) The method as in claim 20, further comprising:

2   determining if an element entry of the descriptor look up table is nested;

8

3   branching to the nested entry;

4   identifying, from the descriptor look up table, a <u>nested</u> series of actions to perform

5   on elements of the nested entry in order to swap bytes of the entry from a first endianness

6   to a second endianness<u>, where the nested series of actions includes linking and convert-</u>

7   <u>ing</u>.

1   26. (Currently Amended) A computer to convert a data structure by a processor, com-

2   prising:

3   means for calling a byte-swapping engine;

4   means for providing a file access data structure as input to the byte-swapping en-

5   gine;

6   means for providing a descriptor look up table to the byte-swapping engine;

7   means for identifying, from the descriptor look up table, a series of actions to per-

8   form on elements of the file access data structure in order to swap bytes of the file access

9   data structure from a first endianness to a second endianness<u>, where the series of actions</u>

10 <u>include at least one of converting, copying, or linking</u>; and

11   means for performing the identified series of actions on the elements of the file

12 access data structure <u>to convert the file access data structure</u>.

1   27. (Previously Presented) The computer as in claim 26, further comprising:

2   means for using as the file access data structure a file having Direct Access File

3   System (DAFS) protocol.

1   28. (Currently Amended) The computer as in claim 26, further comprising:

2   means for determining if the file access data structure is a critical path data struc-

3   ture, where the critical path data structure includes commonly utilized data structures, and

4   if it is the file access data structure is a critical path data structure, perform byte swap op-

5   erations using specific code functions.

1   29. (Currently Amended) The computer as in claim 26, further comprising:

2   means for determining if the file access data structure is a critical path data struc-

3   ture, where the critical path data structure includes commonly utilized data structures, and

4   if it is not the file access data structure is not a critical path data structure, perform byte

5   swap operations on a data structure header.

1   30. (Previously Presented) The computer as in claim 26, further comprising:

2   means for swapping bytes of the data structure as needed, in response to swapping

3   bytes of the file access data structure.

1   31. (Currently Amended) The computer as in claim 26, further comprising:

2   means for determining if an element entry of the descriptor look up table is

3   nested;

4   means for branching to the nested entry;

5   means for identifying, from the descriptor look up table, a <u>nested</u> series of actions

6   to perform on elements of the nested entry in order to swap bytes of the entry from a first

7   endianness to a second endianness<u>, where the nested series of actions includes converting</u>

8   <u>and linking</u>.

1   32. (Currently Amended) A computer readable media, comprising:

2   said computer readable media containing instructions for execution on a processor

3   for the practice of a method for converting a data structure by a processor, the method

4   having the steps of,

5   calling a byte-swapping engine;

6   providing a file access data structure as input to the byte-swapping engine;

7   providing a descriptor look up table to the byte-swapping engine;

8   identifying, from the descriptor look up table, a series of actions to perform on

9   elements of the file access data structure in order to swap bytes of the file access data

10   structure from a first endianness to a second endianness<u>, where the series of actions in-</u>

11   <u>clude at least one of converting, copying, or linking</u>; and

12   performing the identified series of actions on the elements of the file access data

13   structure<u> to convert the file access data structure</u>.

1   33. (Cancelled)

1   34. (Previously Presented) A method of converting elements of a file access data struc-

2   ture from a first endianness to a second endianness by a processor, comprising:

3   determining if the file access data structure is a critical path data structure; and

4   converting the elements from the first endianness to the second endianness using a

5   set of specific code functions if the file access data structure is a critical path data struc-

6   ture.

1   35. (Previously Presented) The method of claim 34, further comprising:

2   converting a header of the file access data structure from the first endianness to

3   the second endianness using a second set of specific code functions if the file access data

4   structure is not a critical path data structure.

1   36. (Previously Presented) The method of claim 34, further comprising:

2   calling a byte swapping engine to convert selected elements of the file access data

3   structure from the first byte order to the second byte order.

1   37. (Previously Presented) A method for converting a first data structure from a to a sec-

2   ond data structure by a processor, the method comprising the steps of:

3   using a descriptor lookup table to provide actions to be performed on each ele-

4   ment of the first data structure; and

5   stepping through the descriptor table and processing each element of the first data

6   structure according to the element's size and action to convert the first data structure into

7   the second data structure.

1   38. (Previously Presented) The method of claim 37, further comprising:

2   using a byte as the data structure.

1   Please add new claims 39 *et al.*

1   39. (New) The method of claim 2, wherein the critical data path structure includes com-

2   monly used data structures.

1   40. (New) The method of claim 2, wherein the critical data path structure is a direct ac-

2   cess file system (DAFS) header data structure.

1   41. (New) The method of claim 2, wherein the specific code functions are designed to

2   rapidly convert any elements of the data structure to the second endianness without using

3   a byte swapping engine.

1   42. (New) The computer-readable medium of claim 15, wherein the critical data path

2   structure includes commonly used data structures.

1   43. (New) The computer-readable medium of claim 15, wherein the critical data path

2   structure is a direct access file system (DAFS) header data structure.

1   44. (New) The computer-readable medium of claim 15, wherein the specific code func-

2   tions are designed to rapidly convert any elements of the data structure to the second en-

3   dianness without using a byte swapping engine.

1 45. (New) The method of claim 34, wherein the critical data path structure includes com-

2 monly used data structures.

1 46. (New) The method of claim 34, wherein the critical data path structure is a direct ac-

2 cess file system (DAFS) header data structure.

1 47. (New) The method of claim 34, wherein the specific code functions are designed to

2 rapidly convert any elements of the data structure to the second endianness without using

3 a byte swapping engine.